**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**«Сибирский государственный университет науки и технологий   
имени академика М.Ф. Решетнева»**

Институт информатики и телекоммуникаций

Кафедры информационно-управляющих систем

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Теория информации

|  |
| --- |
| Определение и исправление ошибок в сообщении. Код хемминга |

Руководитель А.Н. Бочаров

подпись, дата инициалы, фамилия

Обучающийся БПИ22-02, 221219007 И.О. Григорьев

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2024 г.

# ЦЕЛЬ РАБОТЫ

*Закрепление знаний по методам кодирования информации.*

# постановка задачи

Построить код Хемминга для исправления одиночной ошибки и обнаружения двойной ошибки. Код должен предусматривать возможность посылки n сообщений.

Вариант 3: Кол-во сообщ. *N* = 64

1. Рассчитать параметры кода: *n*и , *n*к , n. Вариант задания определяется

преподавателем.  
 2. Привести пример построения 10 кодовых сообщений.  
 3. Показать процедуру исправления ошибки в одной из позиций и

обнаружения двойной ошибки.   
 4. Составить программу, кодирующую и декодирующую кодовую

комбинацию с целью исправления одиночной ошибки и обнаружения двойной

ошибки.

# ХОД РАБОТЫ

**Задание 1.**

Кол-во сообщ. *N* = 64

Разрядность информационной части кода:

*n*и = log2 *N* = log2 64 = 6 бит.

Разрядность контрольной части кода:

*n*к = log2 ((*n*и + 1) + log2 (*n*и + 1)) = log2 (7 + log2 7) ≈ 3,2939 = 4 бит.

Общая разрядность кода: *n* = *n*и + *n*к = 6 + 4 = 10 бит.

**Задание 2.**

Построения расширенного кода Хемминга:

Допустим информационная часть кода будет такой: 010110

*n*и = 6 бит, *n*к = 4 бит, *n* = 10 бит.

Контрольные разряды: 20 = 1, 21 = 2, 22 = 4, 23 = 8.

Макет кода приведен в Таблице 1.

Таблица 1 – Макет кода Хемминга

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 | a4 | a5 | a6 | a7 | a8 | a9 | a10 |
| к1 | к2 | 0 | к3 | 1 | 0 | 1 | к8 | 1 | 0 |

Первая проверка:

a1 + a3 + a5 + a7 + a9 = K1 + 0 + 1 + 1 + 1 = 0.

Число единиц нечётное, значит K1 = 1.

Вторая проверка:

a2 + a3 + a6 + a7 + a10 = K2 + 0 + 0 + 1 + 0 = 0.

Число единиц нечётное, значит K2 = 1.

Третья проверка:

a4 + a5 + a6 + a7 = K3 + 1 + 0 + 1 = 0.

Число единиц чётное, значит K3 = 0.

Четвертая проверка:

a8 + a9 + a10 = K4 + 1 + 0 = 0.

Число единиц нечётное, значит K3 = 1.

Кода приведен в Таблице 2.

Таблица 2 – Код Хемминга

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 | a4 | a5 | a6 | a7 | a8 | a9 | a10 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |

1100101110к5 – кол-во единиц чётное => к5 = 0

11001011100 – расширенный код Хемминга.

10 составленных кодовых сообщений приведены в таблице 3.

Таблица 3 – Коды Хемминга

|  |  |  |
| --- | --- | --- |
| № | Кодовое сообщение | Код Хемминга |
| 1 | 010110 | 11001011100 |
| 2 | 001010 | 01101010100 |
| 3 | 001011 | 11101110110 |
| 4 | 001100 | 00110000100 |
| 5 | 001111 | 11111110100 |
| 6 | 001101 | 10110110110 |
| 7 | 010011 | 11001110110 |
| 8 | 010010 | 01001010100 |
| 9 | 010101 | 10010110100 |
| 10 | 010111 | 11011110110 |

**Задание 3.**

Проверим код на ошибки, это представлено в Таблице 4:

Таблица 4 – Коды Хемминга

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 | a4 | a5 | a6 | a7 | a8 | a9 | a10 | a11 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |

11001011100 – чётное кол-во единиц => ошибки нет или ошибка двойная.

Первая проверка: a1 + a3 + a5 + a7 + a9 = 1 + 0 + 1 + 1 + 1 = 0 => s1 = 0.

Вторая проверка: a2 + a3 + a6 + a7 + a10 = 1 + 0 + 0 + 1 + 0 = 0 => s2 = 0.

Третья проверка: a4 + a5 + a6 + a7 = 0 + 1 + 0 + 1 = 0 => s3 = 0.

Четвертая проверка: a8 + a9 + a10 = 1 + 1 + 0 = 0 => s4 = 0.

Синдром имеет вид S (s4, s3, s2, s1) = 0000 => я сделал правильно, ошибок нет.

Теперь допустим ошибку в 5 разряде, это представлено в Таблице 5:

Таблица 5 – Коды Хемминга

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 | a4 | **a5** | a6 | a7 | a8 | a9 | a10 | a11 |
| 1 | 1 | 0 | 0 | **1** | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | **0** | 0 | 1 | 1 | 1 | 0 | 0 |

Первая проверка: a1 + a3 + a5 + a7 + a9 = 1 + 0 + 0 + 1 + 1 + 0 = 1 => s1 = 1.

Вторая проверка: a2 + a3 + a6 + a7 + a10 = 1 + 0 + 0 + 1 + 0 = 0 => s2 = 0.

Третья проверка: a4 + a5 + a6 + a7 = 0 + 0 + 0 + 1 = 1 => s3 = 1.

Четвертая проверка: a8 + a9 + a10 = 1 + 1 + 0 = 0 => s4 = 0.

Синдром имеет вид S (s4, s3, s2, s1) = 01012 = 510 => ошибка в 5 разряде.

Теперь помимо 5 разряда допустим ошибку в 6 разряде, это представлено в Таблице 6:

Таблица 6 – Коды Хемминга

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a1 | a2 | a3 | a4 | **a5** | **a6** | a7 | a8 | a9 | a10 | a11 |
| 1 | 1 | 0 | 0 | **1** | **0** | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | **0** | **0** | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | **0** | **1** | 1 | 1 | 1 | 0 | 0 |

Первая проверка: a1 + a3 + a5 + a7 + a9 = 1 + 0 + 0 + 1 + 1 = 1 => s1 = 1.

Вторая проверка: a2 + a3 + a6 + a7 + a10 = 1 + 0 + 1 + 1 + 0 = 1 => s2 = 1.

Третья проверка: a4 + a5 + a6 + a7 = 0 + 0 + 1 + 1 = 0 => s3 = 0.

Четвертая проверка: a8 + a9 + a10 = 1 + 1 + 0 = 0 => s4 = 0.

Синдром имеет вид S (s4, s3, s2, s1) = 0011. Синдром не нулевой => ошибка двойная.

**Задание 4.**

Текст программы представлен в Листинге 1.

Листинг 1 – Текст программы

import random

# 10 случайных 6-разрядных двоичных кодов

def generate\_binary\_list(length=6, count=10):

binary\_list = []

for \_ in range(count):

binary\_num = ''.join(str(random.randint(0, 1)) for \_ in range(length))

binary\_list.append(binary\_num)

return binary\_list

# Расширенный код Хемминга

def modified\_hamming\_code(number):

k1 = ((number[0] + number[1] + number[3] + number[4] ).count('1')) % 2

k2 = ((number[0] + number[2] + number[3] + number[5] ).count('1')) % 2

k3 = ((number[1] + number[2] + number[3] ).count('1')) % 2

k4 = ((number[4] + number[5]).count('1')) % 2

result = str(k1) + str(k2) + number[0] + str(k3) + number[1:4] + str(k4) + number[4:6]

if (result.count('1') % 2) == 0:

hamming\_code = result + '0'

else:

hamming\_code = result + '1'

return hamming\_code

# Добавление в коды одиночной ошибки случайным образом

def single\_mistake(code):

code\_list = list(code)

index = random.randint(0, 10)

if code\_list[index] == '1':

code\_list[index] = '0'

else:

code\_list[index] = '1'

single\_mistake\_code = ''.join(code\_list)

return single\_mistake\_code, index+1

# Добавление в коды двойной ошибки случайным образом

def double\_mistake(code):

code\_list = list(code)

indices = random.sample(range(10), 2)

for index in indices:

if code\_list[index] == '1':

code\_list[index] = '0'

else:

code\_list[index] = '1'

double\_mistake\_code = ''.join(code\_list)

return double\_mistake\_code, (indices[0]+1, indices[1]+1)

# Обнаружение ошибок

def mistake\_detection(code):

s1 = ((code[0] + code[2] + code[4] + code[6] + code[8]).count('1')) % 2

s2 = ((code[1] + code[2] + code[5] + code[6] + code[9]).count('1')) % 2

s3 = ((code[3] + code[4] + code[5] + code[6]).count('1')) % 2

s4 = ((code[7] + code[8] + code[9]).count('1')) % 2

s = str(s4) + str(s3) + str(s2) + str(s1)

if ((code.count('1') % 2) == 1):

return s

else:

if int(s, 2) == 0:

return 'Ошибки нет'

else:

return s

# Исправление одиночной ошибки

def single\_mistake\_correction(code, syndrome):

s = int(syndrome, 2)

code\_list = list(code)

if code\_list[s-1] == '1':

code\_list[s-1] = '0'

else:

code\_list[s-1] = '1'

correct\_code = ''.join(code\_list)

return correct\_code

binary\_numbers = generate\_binary\_list()

hamming\_codes\_list = []

for number in binary\_numbers:

hamming\_codes\_list.append(modified\_hamming\_code(number))

single\_mistake\_codes\_list = []

single\_mistake\_indicies = []

for code in hamming\_codes\_list:

single\_mistake\_code, single\_mistake\_index = single\_mistake(code)

single\_mistake\_codes\_list.append(single\_mistake\_code)

single\_mistake\_indicies.append(single\_mistake\_index)

double\_mistake\_codes\_list = []

double\_mistake\_indicies = []

for code in hamming\_codes\_list:

double\_mistake\_code, double\_mistake\_index = double\_mistake(code)

double\_mistake\_codes\_list.append(double\_mistake\_code)

double\_mistake\_indicies.append(double\_mistake\_index)

single\_mistake\_syndrome = []

for code in single\_mistake\_codes\_list:

single\_mistake\_syndrome.append(mistake\_detection(code))

double\_mistake\_syndrome = []

for code in double\_mistake\_codes\_list:

double\_mistake\_syndrome.append(mistake\_detection(code))

single\_mistake\_corrections\_list = []

for i in range(len(single\_mistake\_codes\_list)):

single\_mistake\_corrections\_list.append(single\_mistake\_correction(single\_mistake\_codes\_list[i], single\_mistake\_syndrome[i]))

print("Кодовое сообщение | Код Хемминга")

for i in range(len(binary\_numbers)):

print(binary\_numbers[i], " |", hamming\_codes\_list[i])

print("\n\nОбнаружение и исправление одиночной ошибки")

print("Код Хемминга | Код с 1 ошибкой | Ошибочный разряд | Обнаружение одиночной ошибки | Исправленный код")

for i in range(len(hamming\_codes\_list)):

print(f'{hamming\_codes\_list[i]} | {single\_mistake\_codes\_list[i]} | {single\_mistake\_indicies[i]}'

f' | Вектор ошибки s = {single\_mistake\_syndrome[i]} => ошибка в '

f'{int(single\_mistake\_syndrome[i], 2)} разряде | {single\_mistake\_corrections\_list[i]}')

print("\n\nОбнаружение двойной ошибки")

print("Код Хемминга | Код с 2 ошибкой | Ошибочные разряды | Обнаружение двойной ошибки")

for i in range(len(hamming\_codes\_list)):

print(f'{hamming\_codes\_list[i]} | {double\_mistake\_codes\_list[i]} | {(double\_mistake\_indicies[i])[0]} '

f'{(double\_mistake\_indicies[i])[1]} |Вектор ошибки s = {double\_mistake\_syndrome[i]} '

f'!= 0 => ошибка двойная')

Результаты работы программы показаны на Рисунке 1.

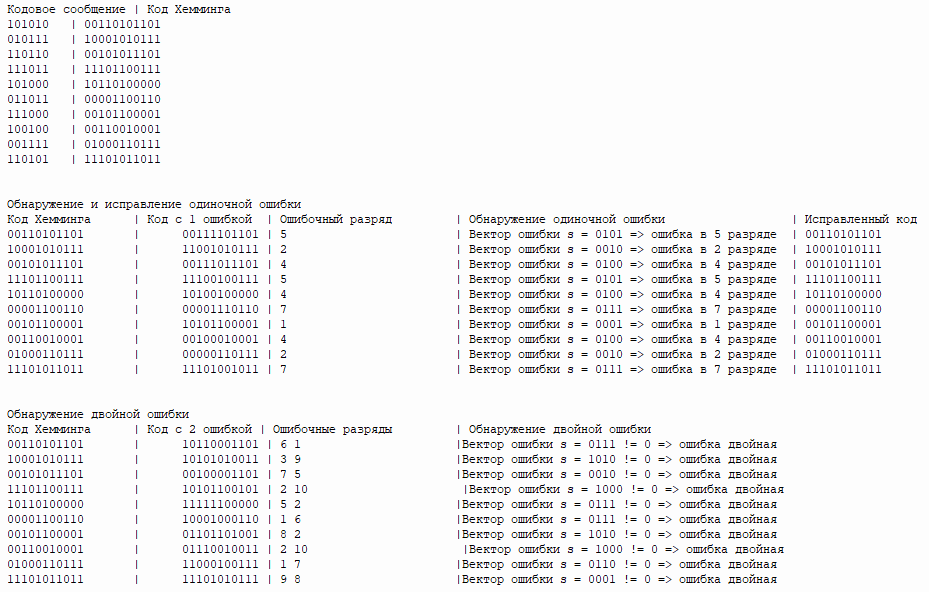


Рисунок 1 – Результаты

# ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое корректирующий код?

Корректирующие или помехоустойчивые коды — это коды, позволяющие обнаруживать и исправлять ошибки, происходящие в процессе передачи из-за влияния помех.

1. Как обнаруживаются ошибки при передаче информации с помощью корректирующего кода?

Примером корректирующих кодов являются коды с контролем по чётности (нечётности). Суть данных кодов состоит в следующем: к исходной кодовой комбинации добавляется 1 или 0 таким образом, чтобы сумма разрядов кодовой комбинации всегда была чётной или нечётной. Сбой любого одного символа нарушит условие четности, и ошибка будет обнаружена.

1. Как рассчитать параметры корректирующего кода?

К основным характеристикам корректирующих кодов относятся:

- число разрешённых и запрещённых кодовых комбинаций;

- избыточность кода;

- минимальное кодовое расстояние;

- число обнаруживаемых и исправляемых ошибок;

- корректирующие возможности кода.

Для блочных двоичных кодов, с числом символов в блоках равным n, общее число возможных кодовых комбинаций определяется значением N0 = 2n

Число разрешённых кодовых комбинаций при наличии nи информационных разрядов в первичном коде равно Nр=2 nи

Очевидно, что число запрещенных комбинаций равно Nз=N0−N р=2 n−2 nи

1. Как определить кодовое расстояние?

Количество разрядов (символов), которыми отличаются две кодовые комбинации, можно принять за кодовое расстояние d между ними. Для определения этого расстояния нужно сложить две кодовые комбинации по модулю 2 и подсчитать число единиц в полученной сумме.

1. Как определить минимальное кодовое расстояние?

Расстояние между различными комбинациями некоторого конкретного кода могут существенно отличаться. Поэтому особую важность для характеристики корректирующих свойств кода имеет минимальное кодовое расстояние dmin, определяемое при попарном сравнении всех кодовых комбинаций, которое называют расстоянием Хемминга.

1. Как строится код Хемминга?

Построение кода Хемминга начинается с построения его макета. Первым шагом определяется число информационных, контрольных разрядов и общая разрядность кода. Очень часто задается количество информационных комбинаций (то есть количество разрешенных комбинаций) на основании которпых определяется число информационных разрядов nи. Код Хемминга может обнаружить и исправить одиночную ошибку, следовательно, его минимальное кодовое расстояние dmin = 3. Исходя из количества информационных разрядов определяется количество контрольных разрядов nк. Общая разрядность кода n определяется какой-либо n = nи + nк. Номера позиций контрольных разрядов определяются по закону 2i , где i = 0, 1, 2, 3, ...

1. Как проводится процедура декодирования кода Хемминга?

В процессе декодирования производится вычисление схем проверок и строится вектор ошибки — синдром S (sк, …, s2, s1). Вычисление схем проверок проводится суммирование по модулю 2 разрядов кодовой комбинации, входящих в схему проверок. В результате первой проверки определяется младший разряд синдрома — s1. В результате последней проверки определяется старший разряд синдрома - sк. Если принятая комбинация содержит ошибку, то в векторе образуется число, которое указывает на номер ошибочной позиции в кодовом векторе.

1. Что такое пропускная способность канала связи?

Пропускная способность канала связи - это максимальная скорость, с которой информация может быть передана через канал связи.

1. В чем суть теоремы Шеннона для дискретного канала с помехами?

Суть теоремы Шеннона заключается в том, что существует предел пропускной способности канала, при котором информация может быть передана практически без ошибок, несмотря на шум. Этот предел определяется как "Пропускная способность Шеннона", и он связан с уровнем шума в канале и его шириной полосы.

1. Как описать двоичный канал?

Двоичный канал - это канал связи, в котором передаются двоичные данные (0 или 1). Этот канал может подвергаться различным типам шумов, что может приводить к ошибкам при передаче данных.

1. Как определить пропускную способность двоичного канала?

Пропускную способность двоичного канала можно рассчитать по теореме Шеннона.

1. Что представляет собой вектор ошибок при обнаружении и исправлении ошибок в сообщении?

Вектор ошибок - это вектор, который показывает, в каких местах сообщения произошли ошибки. В векторе ошибок единица указывает на ошибку в соответствующей позиции, а ноль означает отсутствие ошибки.

# ВЫВОДЫ

В ходе выполнения лабораторной работы мной были закреплены знания по методам кодирования информации.